Processor-Sharing Internet of Things Architecture for Large-scale Deployment

Qianhe Meng[1] , Han Wang[1] , Chong Zhang[1,2] , Yihang Song[1] , Songfan Li[3] ,
Li Lu[1] , and Hongzi Zhu[4]

[1]University of Electronic Science and Technology of China [2]Southwest Petroleum University
[3]The Hong Kong University of Science and Technology [4]Shanghai Jiao Tong University

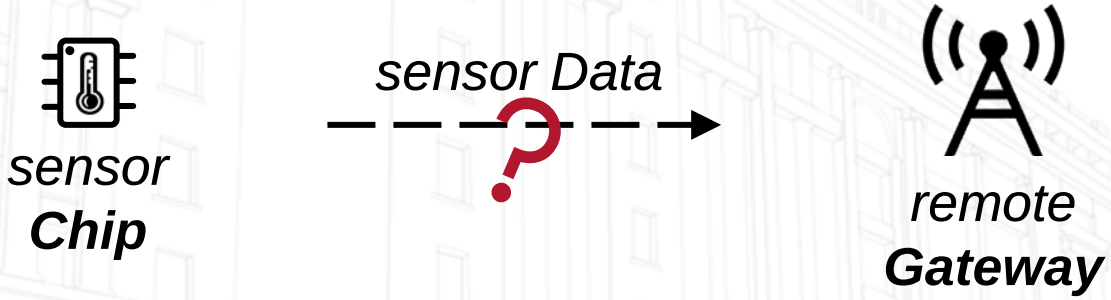
Background

Sensor chips are keen to be **deployed** at a **large-scale.**

scenario #1: wildfire detection

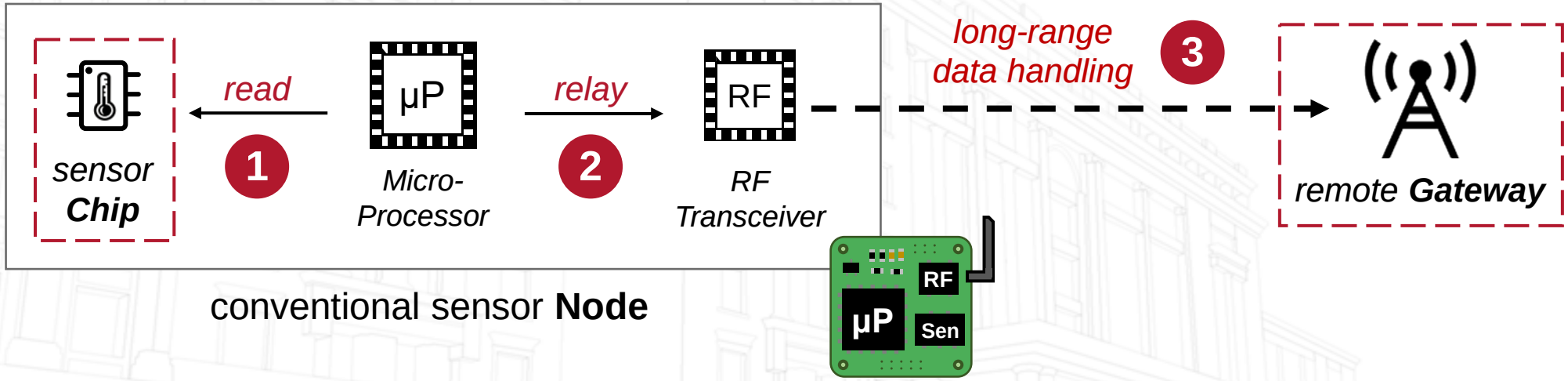
scenario #2: structural health monitoring

Status Quo (1/3)



Status Quo (2/3)

Embedded Solution: Embedding each sensor chip into a full-fledged embedded device.



conventional sensor **Node**

Status Quo (3/3)

Nodes are **EXPENSIVE**
for large-scale deployment

Unaffordable **Manufacturing** Cost

full-fledged with μP and RF transceiver
→ more than $10 per node!

Unaffordable **Maintenance** Cost

power-hungry RF components
→ frequent battery changes!

Common Sense

Simplify nodes
for the ease of large-scale deployment.

Current Effort (1/4)

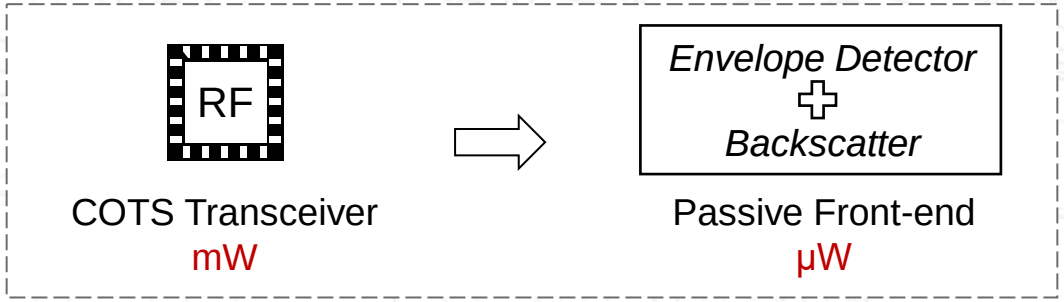
Shifting node's functionality **to** the remote gateway.

sensor **Node** remote **Gateway**

Current Effort (2/4)

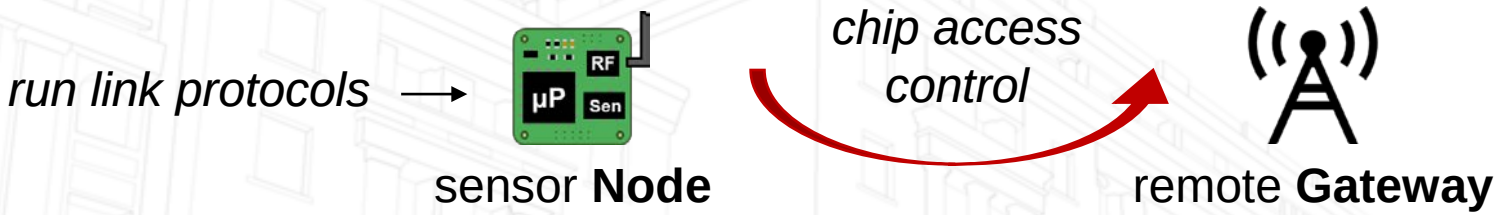
Direction #1: communication offload *(e.g., backscatter)*

Drawback: Two-way communication range **limited to ~30 meters.**

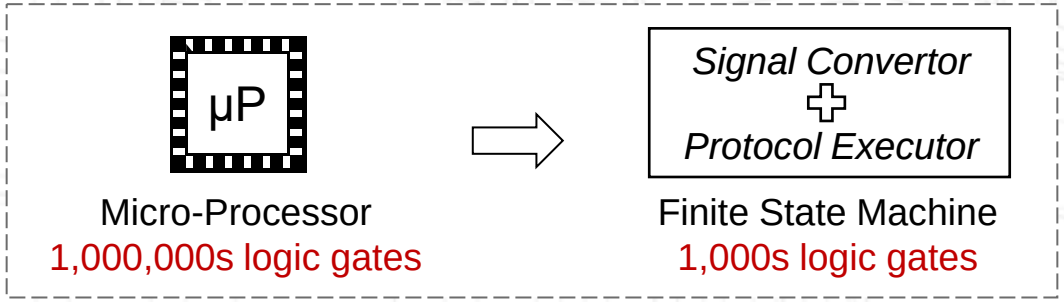


Current Effort (3/4)

Direction #2: computation offload *(e.g., processor-free)*



Drawback: Comm. overheads increased, thus network **scalability sacrificed.**



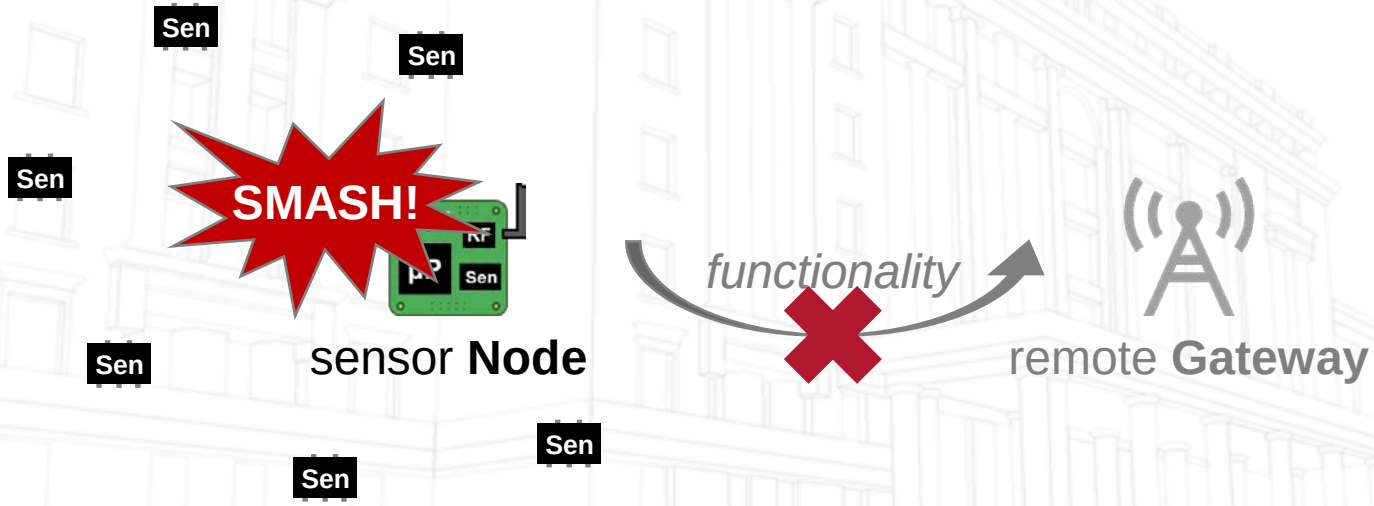
Current Effort (4/4)

Insight: Current effort **trades** the network performance **for** the simplicity of sensor nodes. → More gateways to be deployed in trade-off!

Deployment costs are **INCREASED** at the **SYSTEM-LEVEL!**

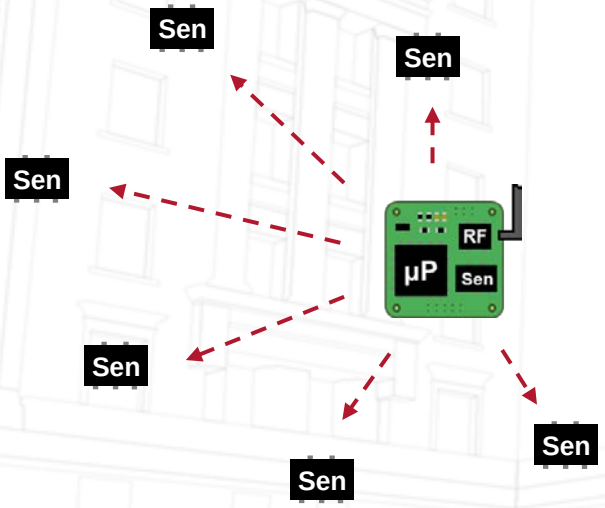
Motivation (1/2)

Smash a full-fledged sensor node into the AIR
for **broader sensor-chip coverage.**



Motivation (2/2)

Could a processor access its **neighboring** sensor **chips** with **negligible** overheads?



aggregated as a sensor **Cluster**

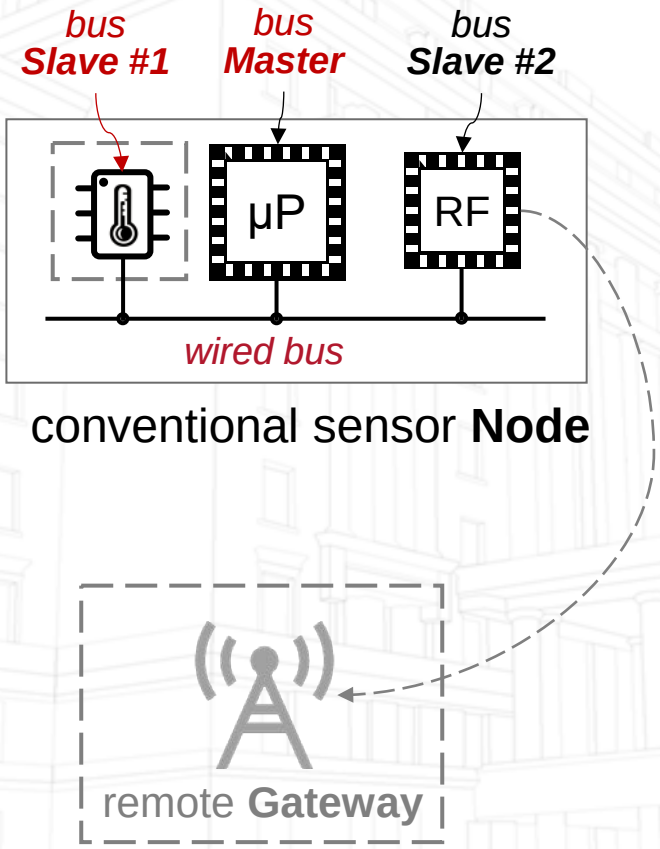
deployment costs ↓

network scalability ↑

remote **Gateway**

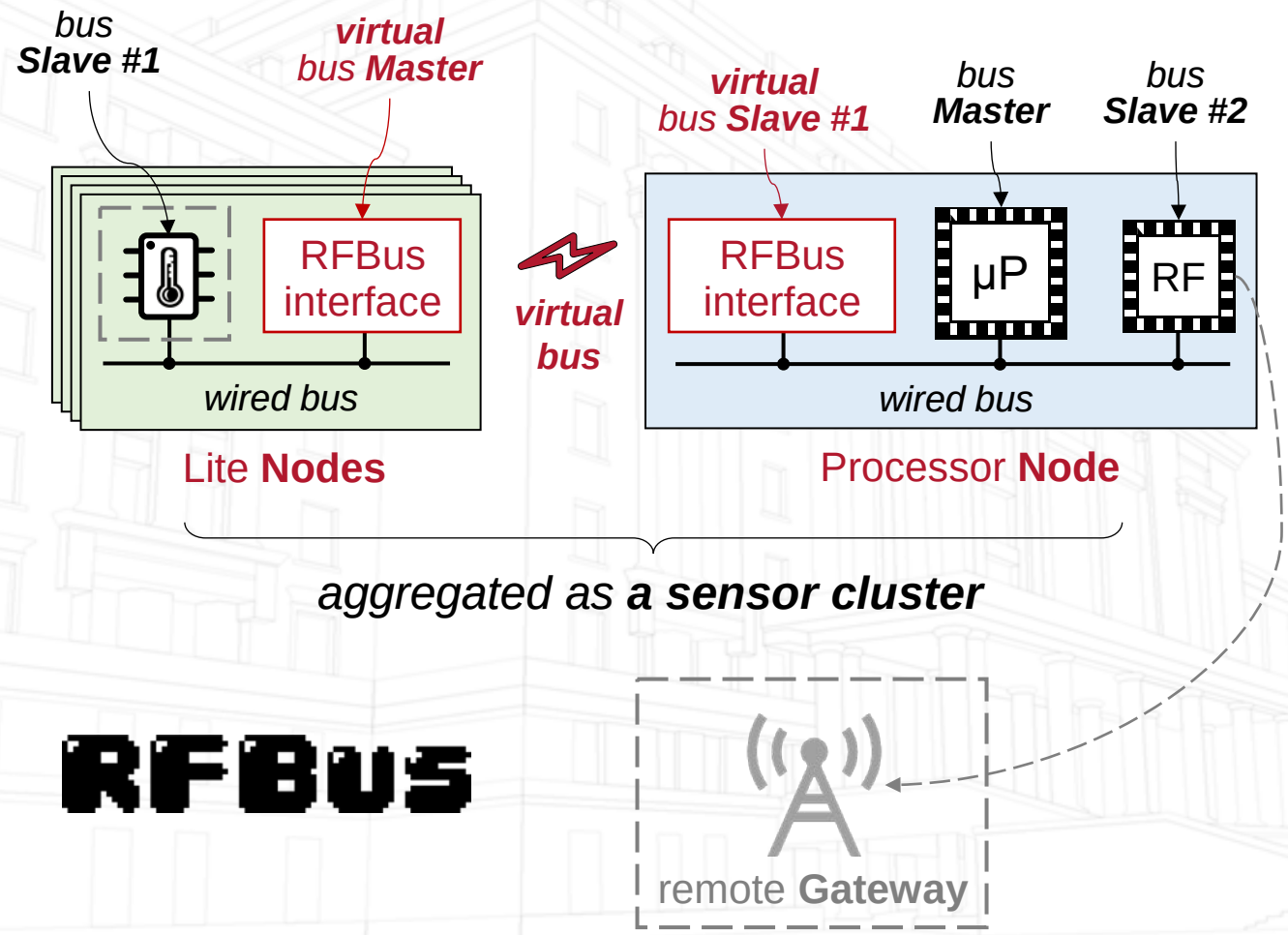
Basic Idea (1/3)

Basic Knowledge: Chips are inter-connected through **computer bus**.



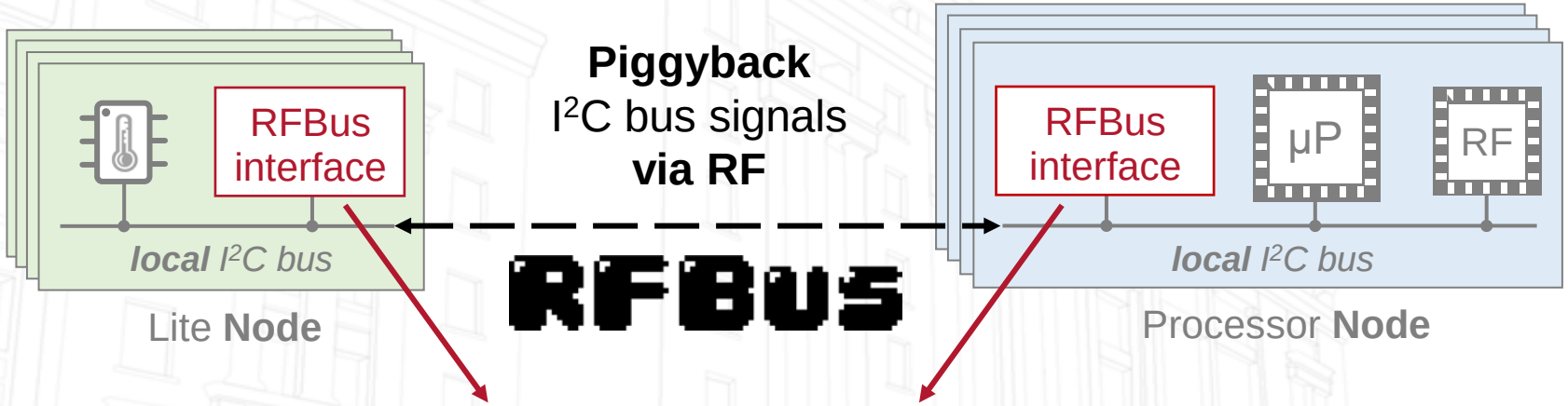
Basic Idea (2/3)

Basic Idea: Constructing a ***virtual*** bus among chips.



Basic Idea (3/3)

Sweet Spot: Inherit link-layer services from I²C protocol transparently, *including chip address, anti-collision and reliable delivery*.



***fully backward compatible** with the I²C specification*

A chip-oriented multi-to-multi RF network
powered by I²C.

Design #1: RF Open-Drain ***(PHY layer)***

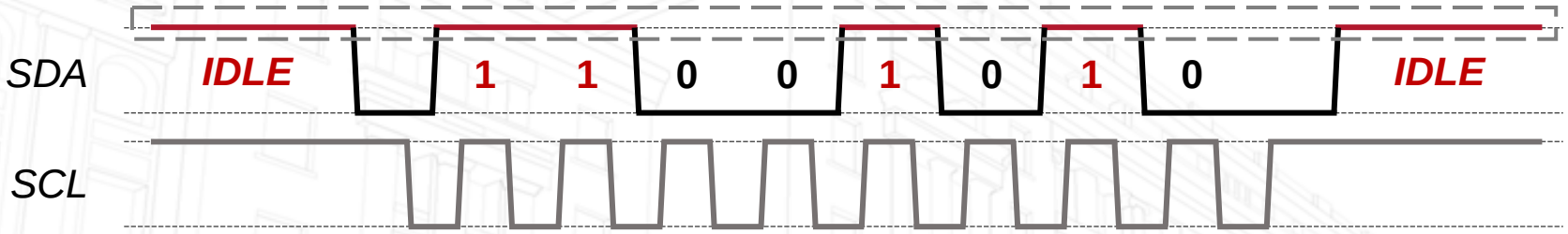
Design #2: RFBus Front-End ***(PHY layer)***

Design #3: Half-duplex RF signaling ***(link layer)***

Design #4: Multi-to-Multi Chip Networking ***(link layer)***

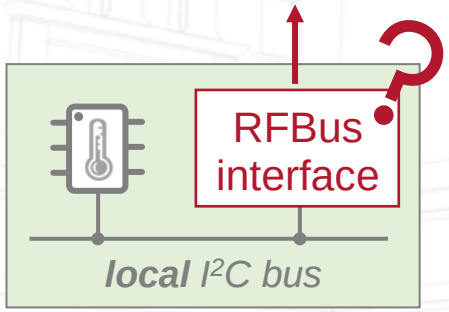
Design > #1 RF Open-Drain (1/2)

Problem: I²C express **trinary** bus information through **binary** voltage-level.



ambiguous
high-SDA

Lite node asks:
*'Does the input high-SDA represents **logic '1'** or **IDLE**?'*

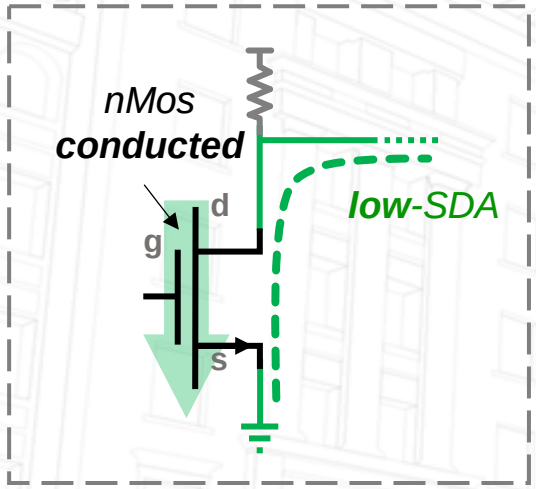


Lite **Node**

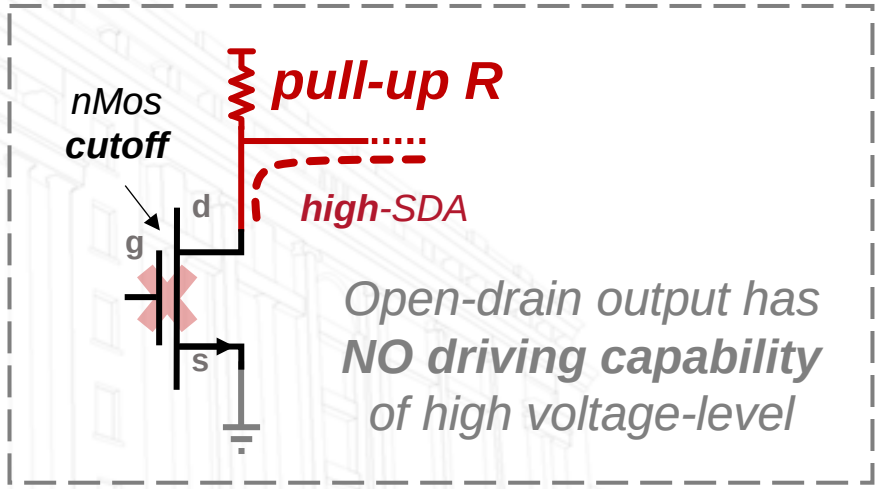
if logic '1'
piggyback it via RF;
else (IDLE)
keep silent (release I²C bus);

Design > #1 RF Open-Drain (2/2)

Insight: Trinary I²C bus information is expressed by binary nMos control.



SDA expresses **logic '0'**

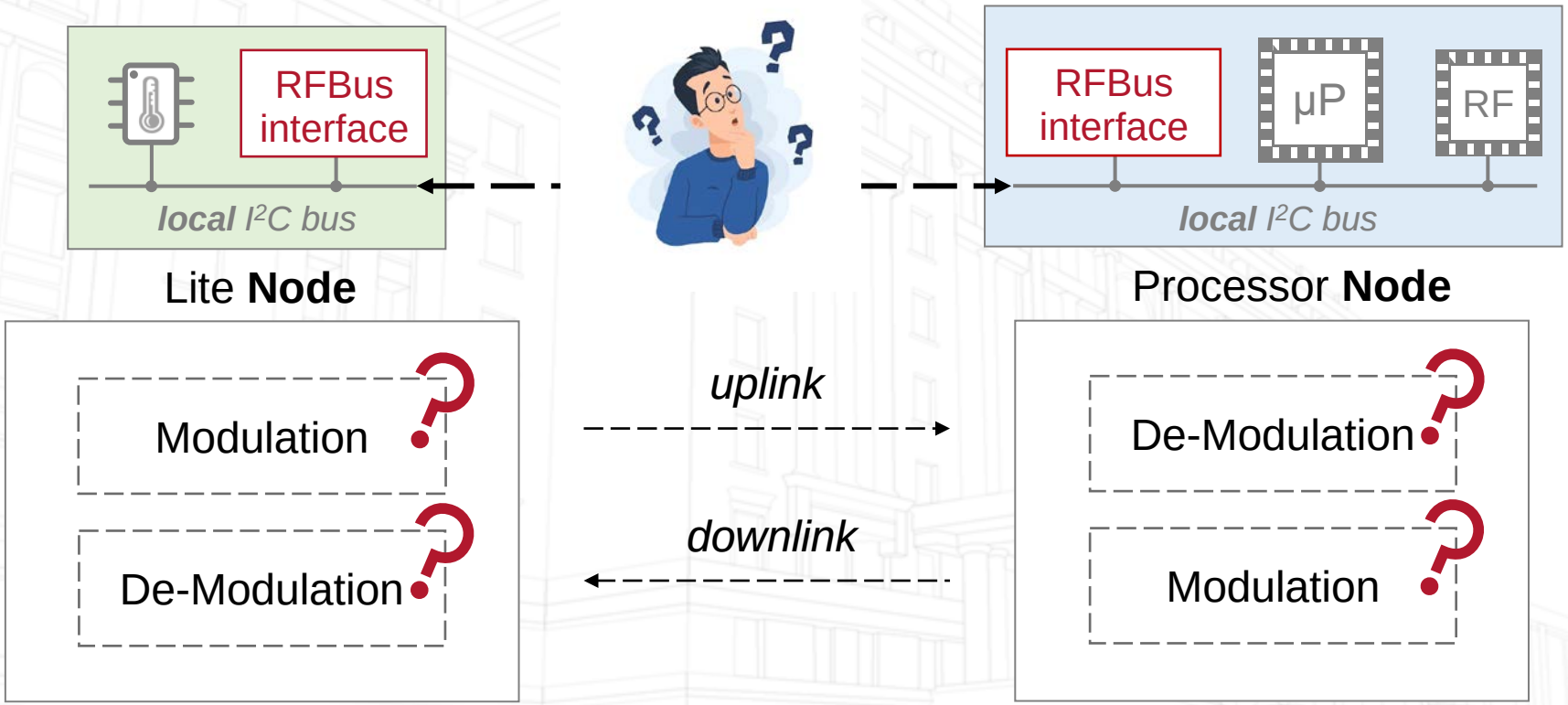


SDA expresses **logic '1'** or **IDLE**

Solution: Piggyback ***low*-SDA** via **RF** only.

Design > #2 RFBus Front-End (1/5)

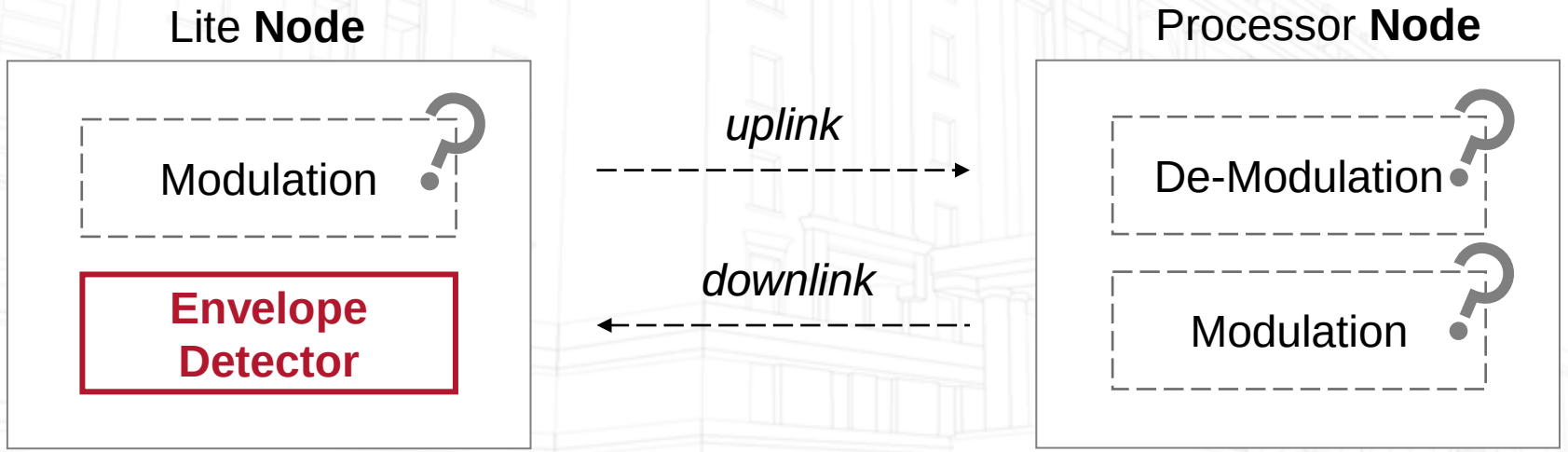
Principle: Inter-node communication with negligible overheads.



low-cost, low-power & fit-for-purpose performance

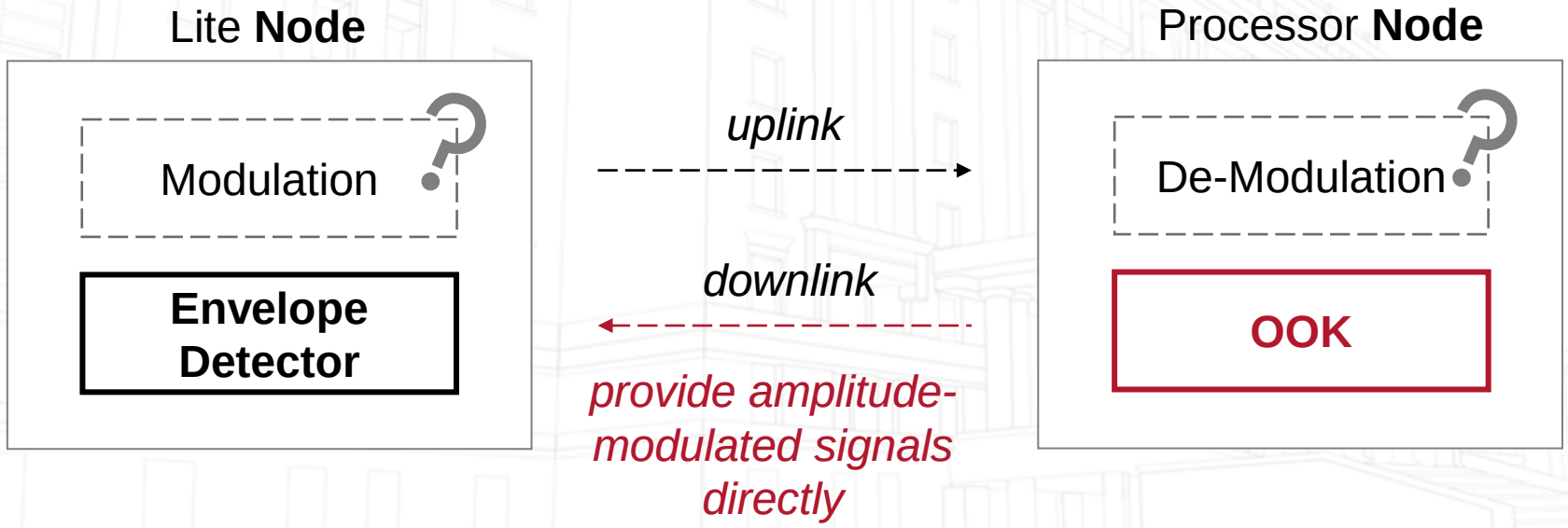
Design Concern: Lite node is RFBus slave, thus must listen to the RFBus continuously.

Ultra-low power RX at lite node



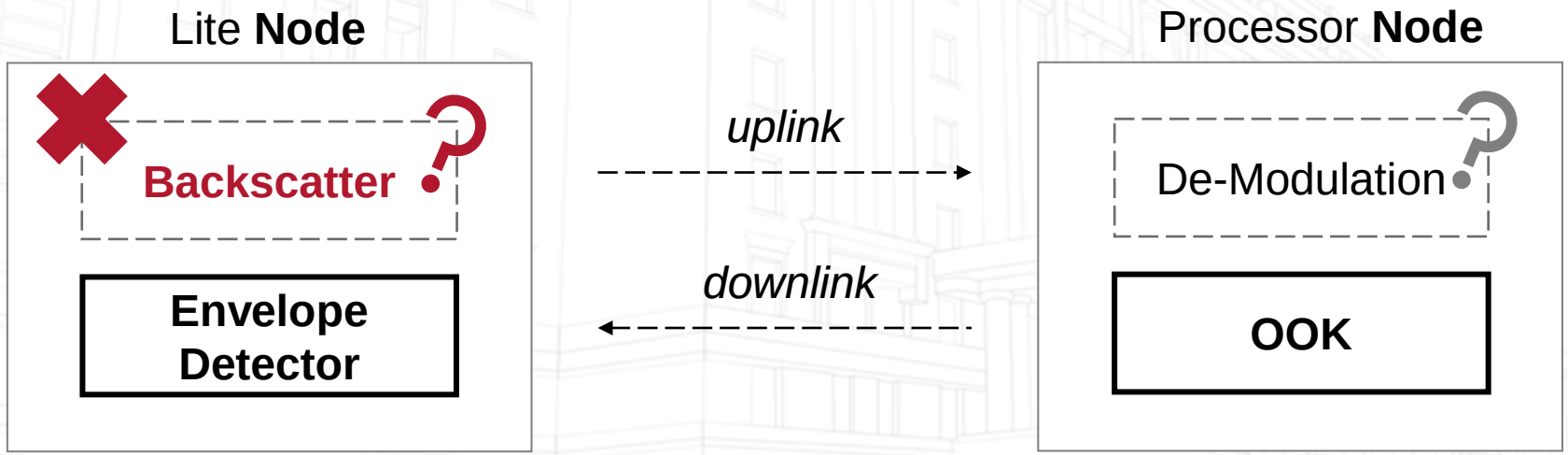
Design Concern: Envelope detector demodulates according to RF signal energy.

OOK TX at processor node



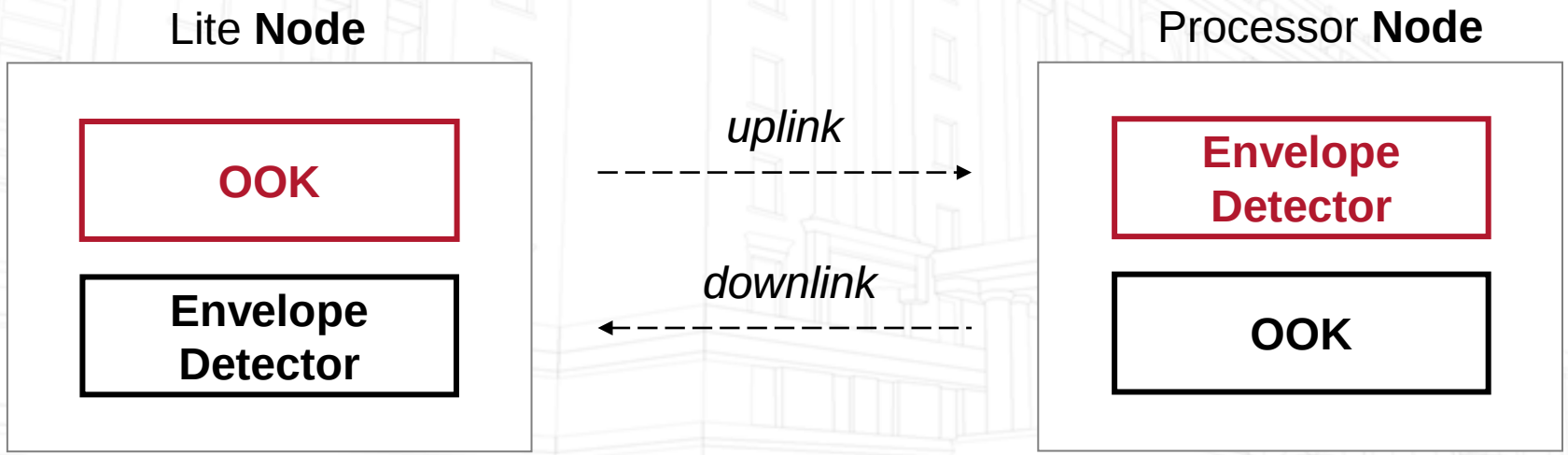
Insight: Backscatter is not suitable for RFBus.

- Backscatter calls for sensitive RX → *deployment costs of processor node increased*.
- Backscatter causes unequal two-way communication range → *inefficient energy utilization*.



Design > #2 RFBus Front-End (5/5)

A **symmetric, bidirectional** RF chain
between the processor and lite nodes.

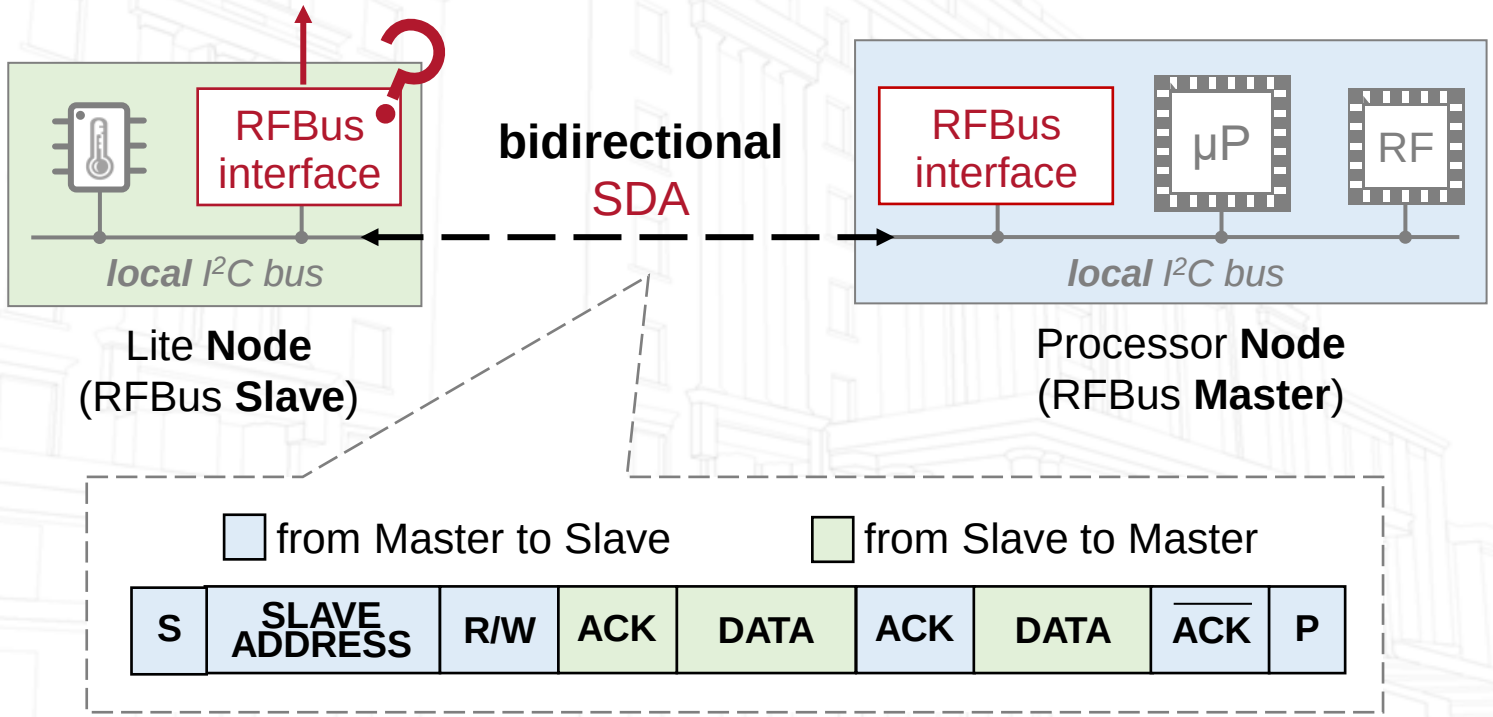


identical** two-way communication **range

Design > #3 Half-Duplex RF Signaling (1/2)

Problem: Rhythm of bidirectional I²C communication is **agnostic** to lite node.

Lite node asks:
'Should I piggyback current SDA symbol?'

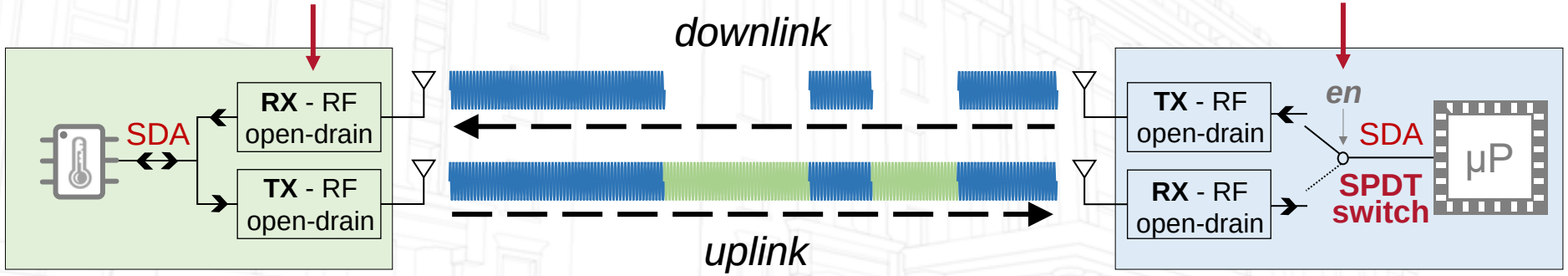


Design > #3 Half-Duplex RF Signaling (2/2)

Solution: Frequency division multiplexing without protocol parsing at lite node.

RFBus replies:
*'Don't worry, just piggyback **all** SDA symbols.'*

RFBus commands:
'Hey, only load SDA symbols generated by sensor chip to your local SDA!'

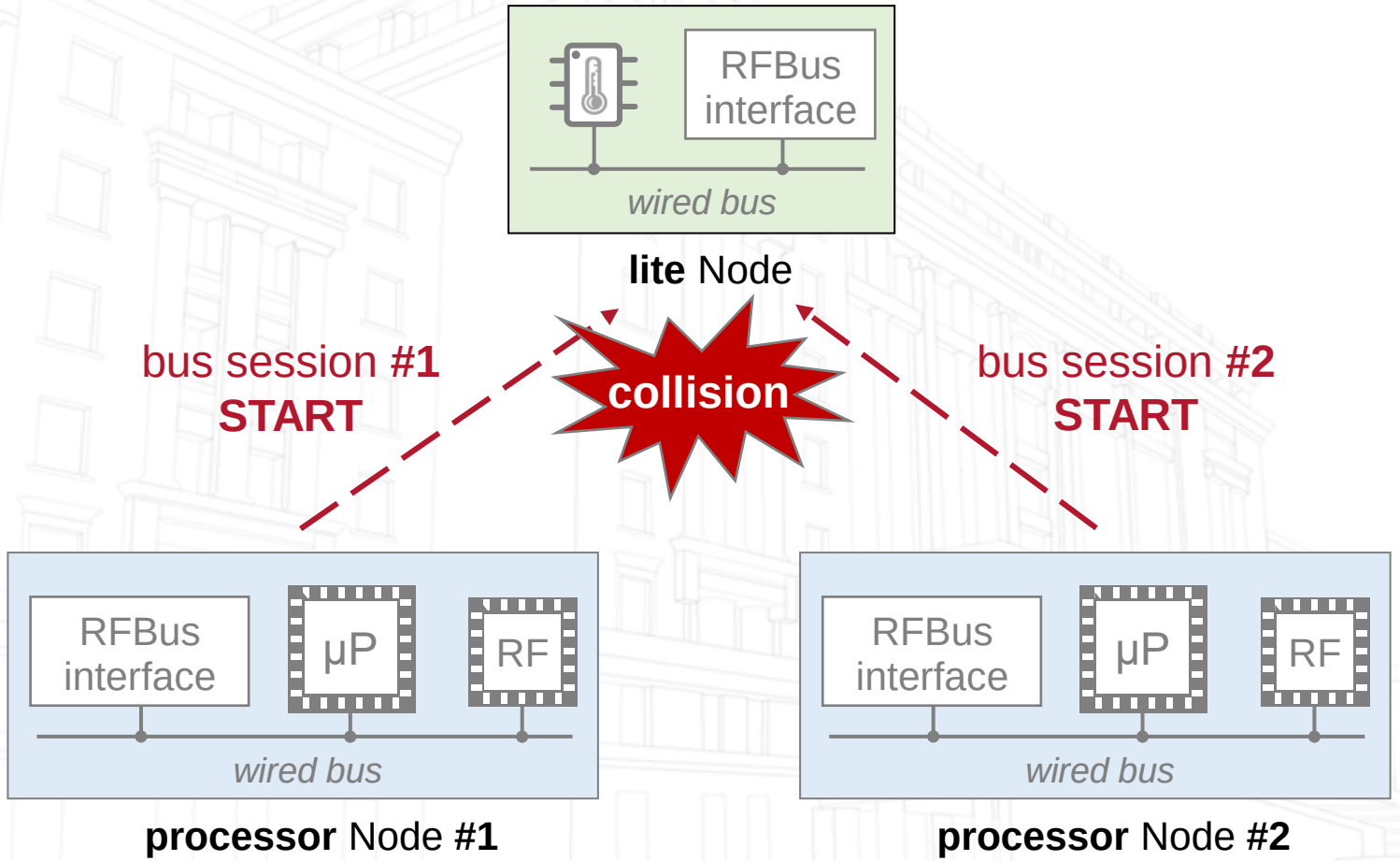


Lite **Node**
(RFBus **Slave**)

Processor **Node**
(RFBus **Master**)

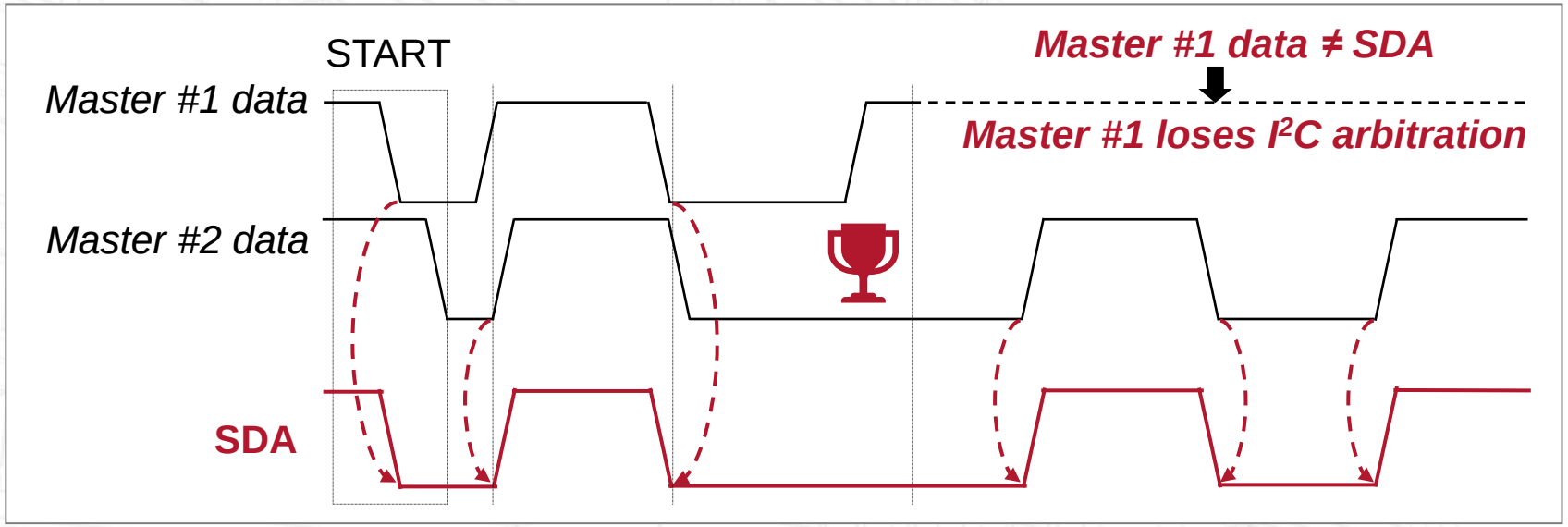
Design > #4 Multi-to-Multi Chip Networking (1/5)

Problem: Bus collisions among multiple processor nodes.



Design > #4 Multi-to-Multi Chip Networking (2/5)

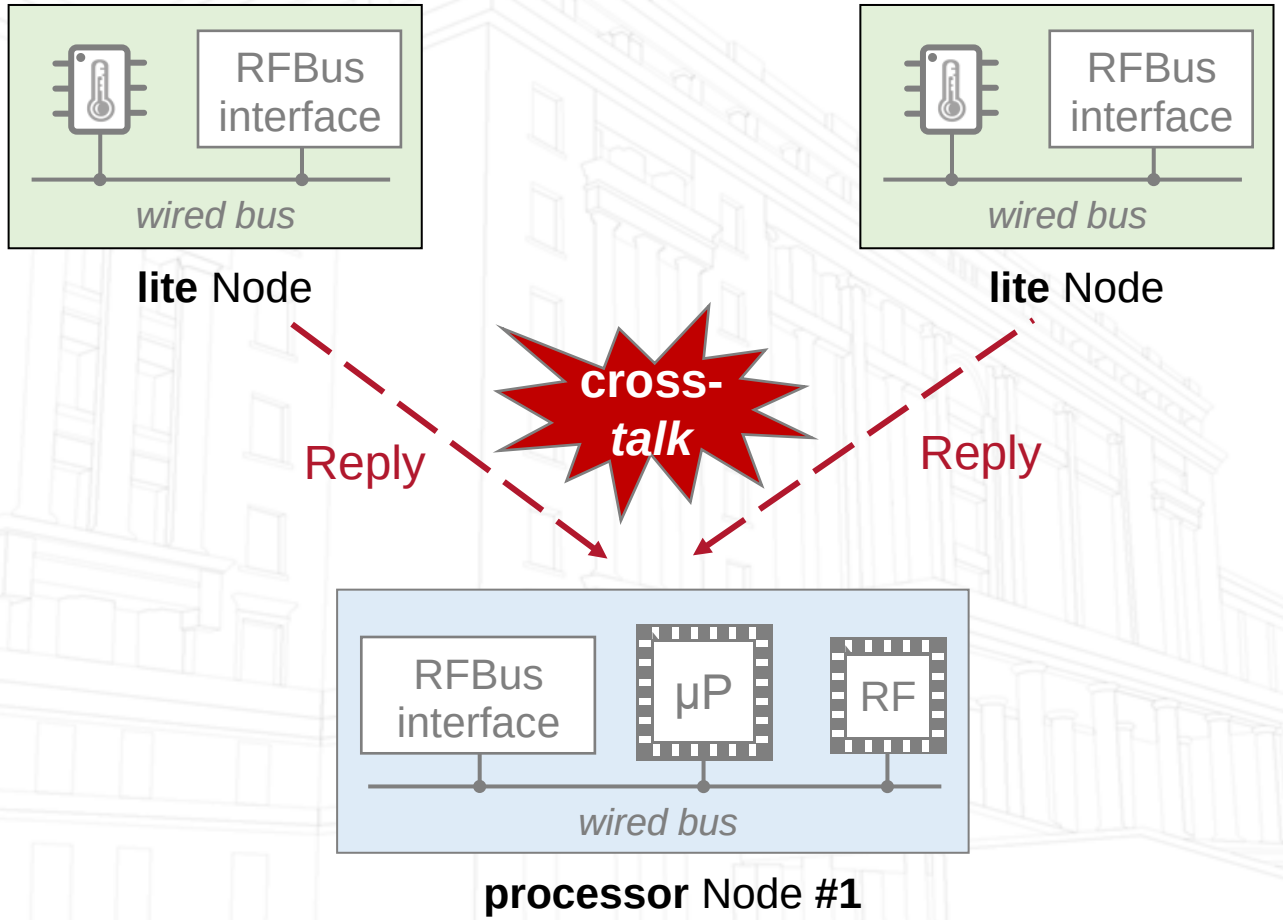
Naturally resolved by I²C arbitration.



The principle of I²C arbitration shown in a two-master case.

Design > #4 Multi-to-Multi Chip Networking (3/5)

Problem: Cross-talk among multiple lite nodes.

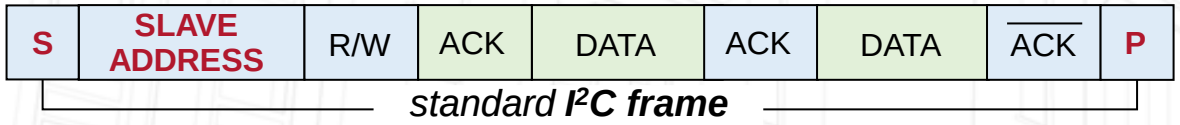


Design > #4 Multi-to-Multi Chip Networking (4/5)

Intuitive Solution: Control lite node's RF by parsing I²C protocol.

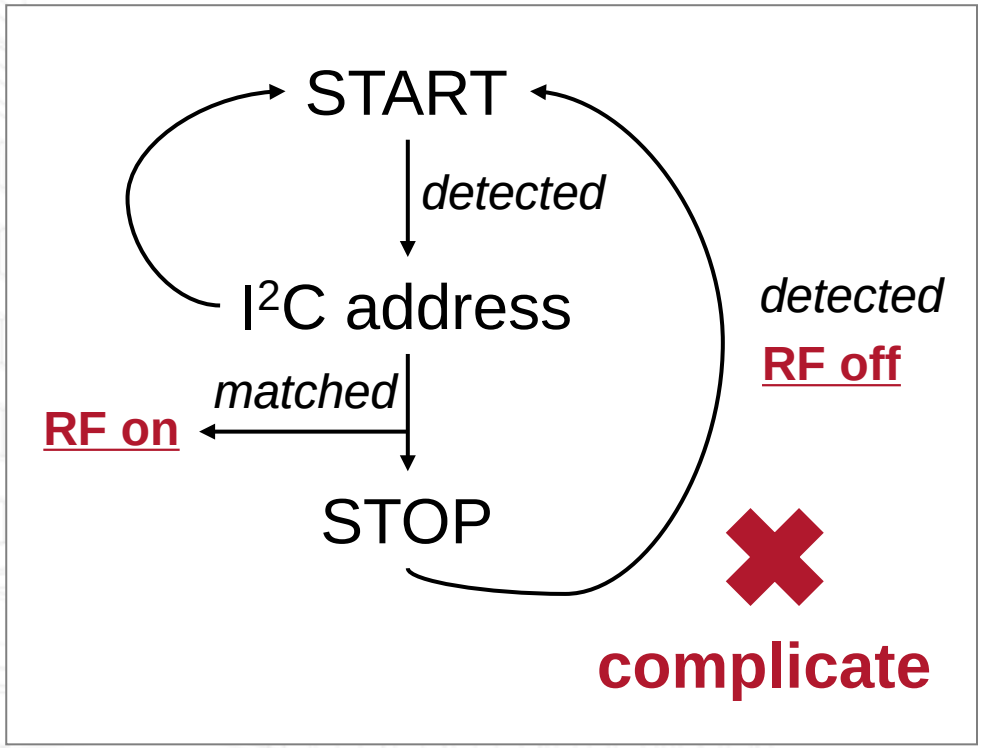
from processor node **to** lite node

from lite node **to** processor node



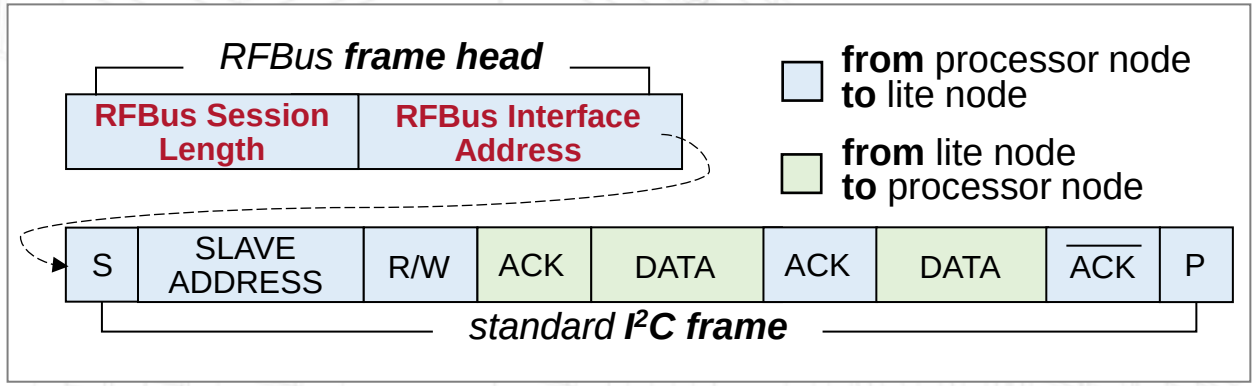
7-bit I²C address =
4-bit Manufacturer ID +
3-bit hardware-defined address

✖ **Only 2^3 = 8** *usable addresses*



Design > #4 Multi-to-Multi Chip Networking (5/5)

Our Solution: Configure lite node's RFBus interface before each I²C session.

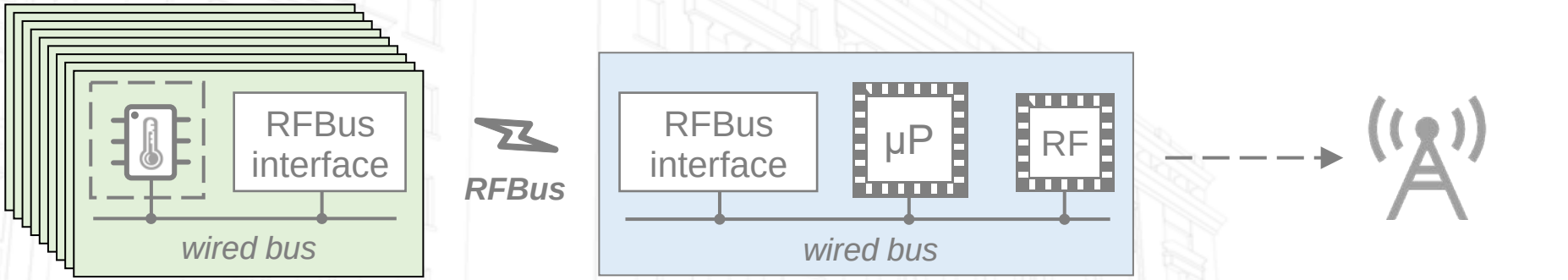


RFBus frame structure

usable I²C address ↓ 2^3 × ↙ 2^8 *RFBus address* = **2048** *usable addresses in RFBus*

Evaluation > Outline

part1 **RFBus Network**
> Inter-Node Communication Range
> Network Throughput
> Task Throughput



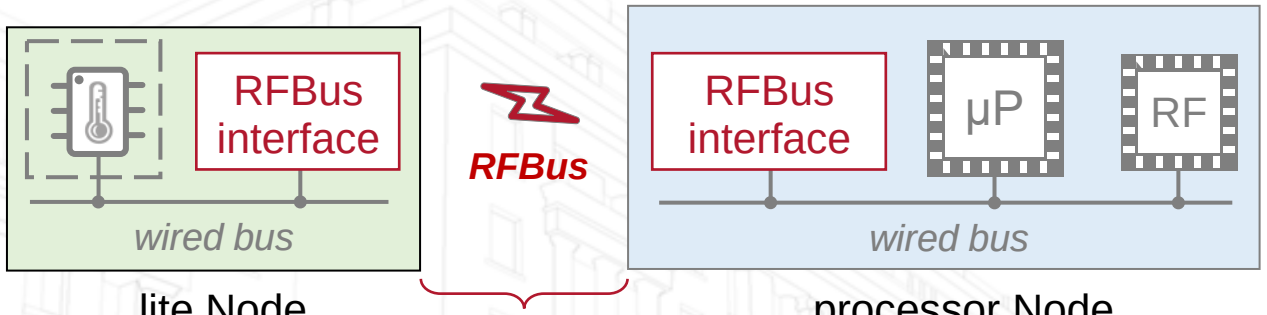
9 lite **Nodes**

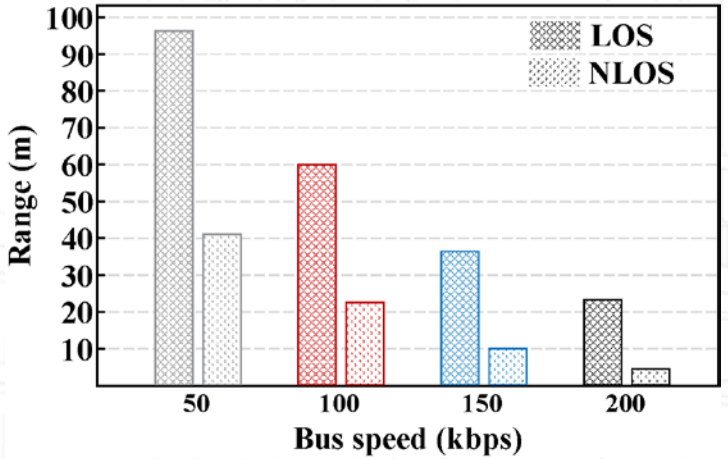
1 processor **Node**

remote **Gateway**

part2 **Processor-Sharing Architecture**
> Power Consumption
> Response Time
> Manufacturing Cost

Setup: passive RX ; 17dBm+2dBi TX



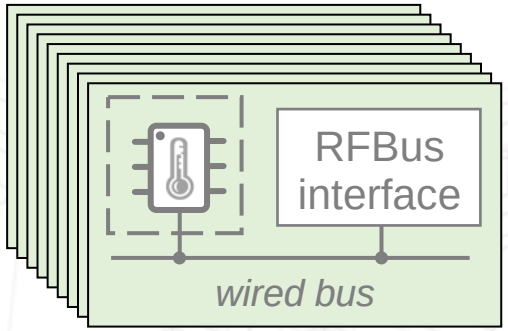


inter-node communication range

96m
(50 kbps; LOS)

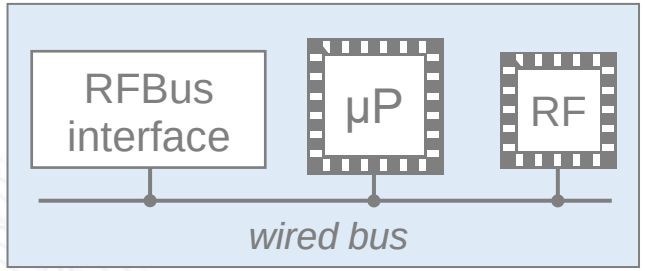
41m
(50 kbps; through door)

Evaluation > RFBus Network > Network and Task Throughput

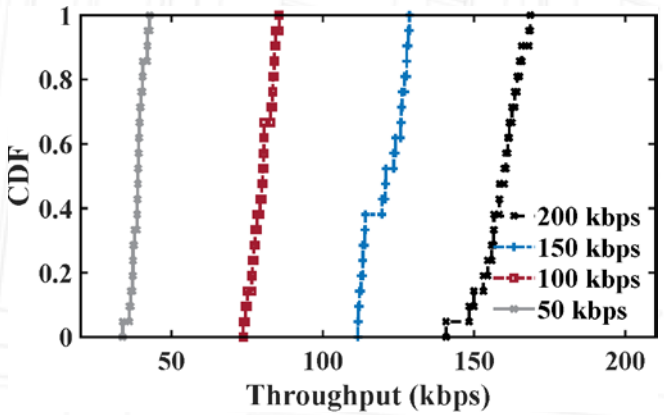


9 lite **Nodes**

read sensor chips
via RFBus



1 processor **Node**



*Limited by the **conducting frequency** of RF Schottky diode*

168 kbps *(max)*
network throughput

Strategy: ***querying***

50 kbps	100 kbps	150 kbps	200 kbps
169 Hz	**339** Hz	**452** Hz	**678** Hz

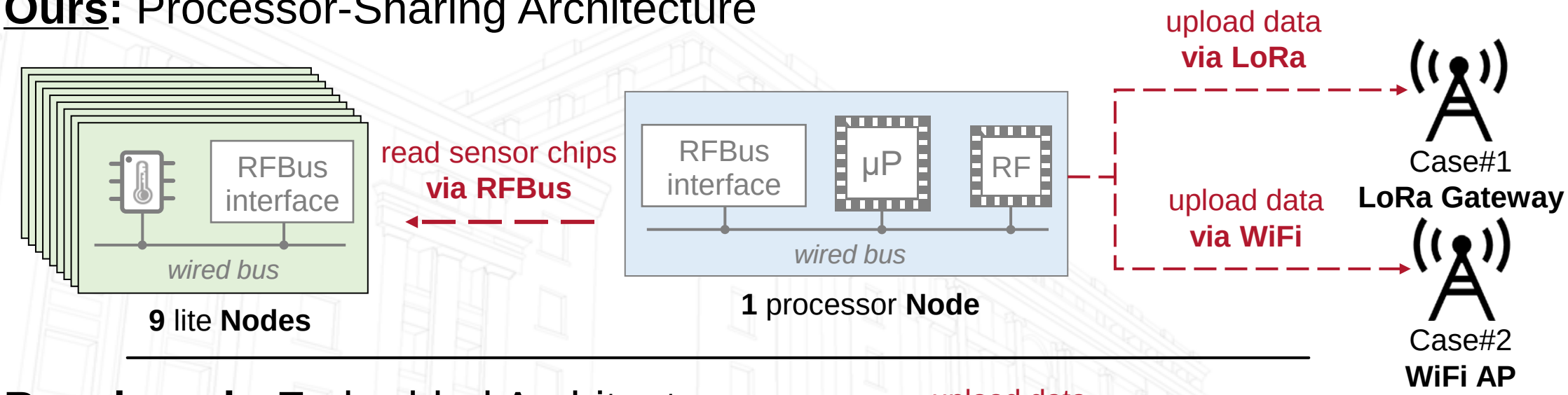
Strategy: ***polling***

50 kbps	100 kbps	150 kbps	200 kbps
218 Hz	**403** Hz	**550** Hz	**826** Hz

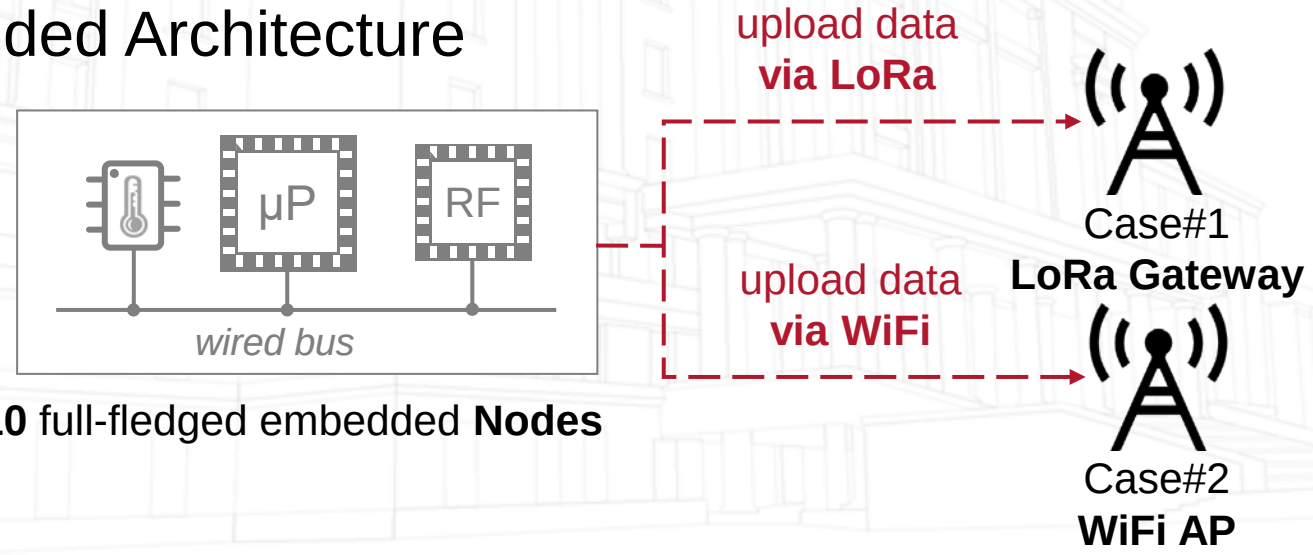
*A processor node reads these **30 sensor chips** up to **826 times** in pipeline within **1 second**.*

24.8 kHz *(max)*
task throughput

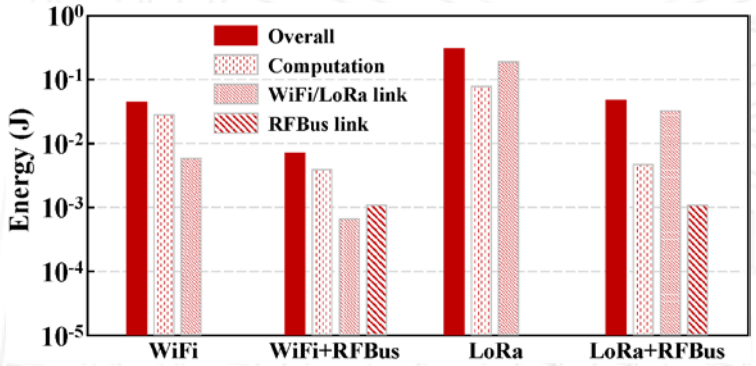
Ours: Processor-Sharing Architecture



Benchmark: Embedded Architecture



Evaluation > P.S. Architecture > Power, Real-time & Cost

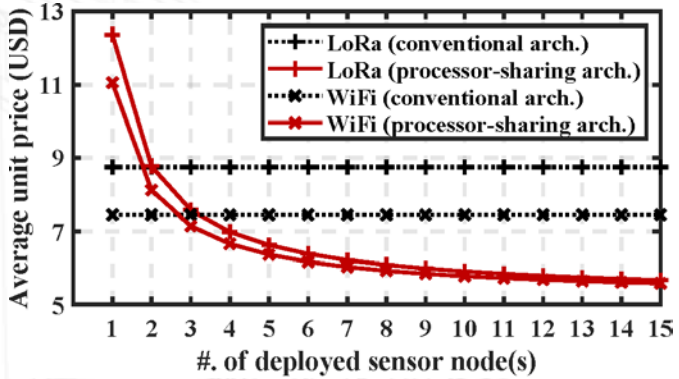


Node's **power consumption** *compared with:*

- *embedded* ***WiFi*** *node* **6.09 ×**
- *embedded* ***LoRa*** *node* **6.69 ×**

WiFi		LoRa	
embedded architecture	P.S. architecture	embedded architecture	P.S. architecture
364 ms	**241 ms**	**1029 ms**	**300 ms**

Response time comparison.

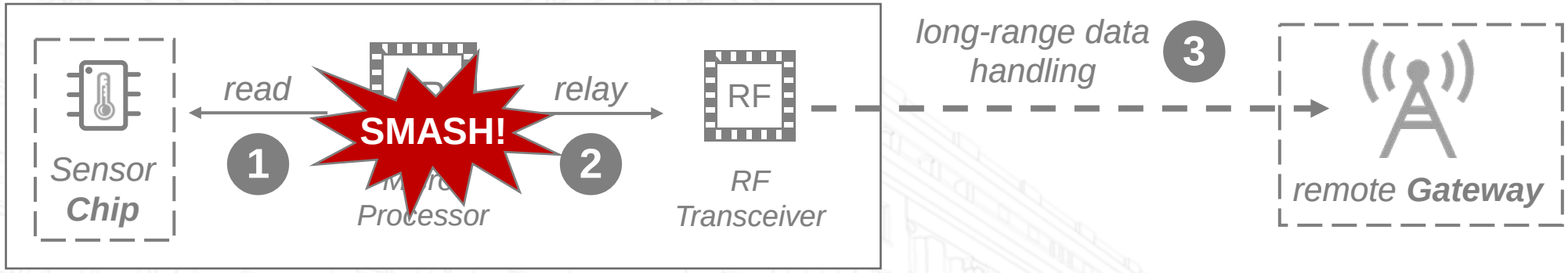


Node's **manufacturing cost** *compared with:*

- *embedded* ***WiFi*** *node* **23.5%**
- *embedded* ***LoRa*** *node* **33.5%**

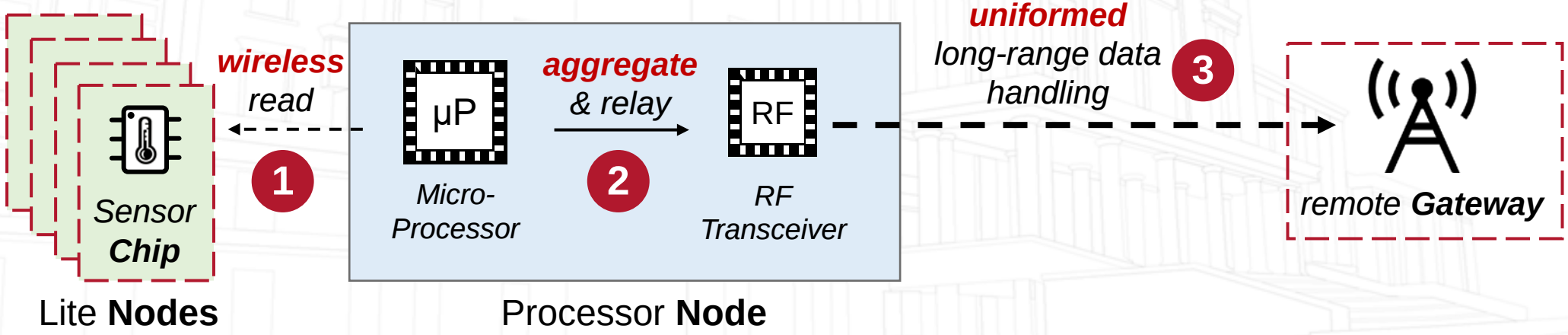
Overview

Embedded Solution:



conventional **sensor Node**

Processor-Sharing Solution:



Lite **Nodes**

Processor **Node**

Processor-Sharing Internet of Things Architecture for Large-scale Deployment

Qianhe Meng

qianhe@std.uestc.edu.cn

more at https://mqhyes.github.io/